11-1994

A Load Distribution through Competition for Workstation Clusters

Kam Hong SHUM
Singapore Management University, khshum@smu.edu.sg

Muslim Bozyigit

A Load Distribution Through Competition for Workstation Clusters*

Kam Hong Shum and Muslim Bozyigit†
University of Cambridge, Computer Laboratory,
New Museums Site, Cambridge CB2 3QG, UK
khs1001@cl.cam.ac.uk

Abstract

The aim of this work is to develop a competition driven solution approach for load distribution in distributed computing system (DCS) environments. The subject DCS is composed of a set of workstation clusters. The study deals with concurrent applications, but assumes the existence of the independent tasks executing on individual workstations as well.

Akin to conventional load balancing algorithms, the approach considers two phases; the partitioning phase and the mapping phase. Each phase is based on the application and the DCS data which is translated into market data, mainly *price*. The *price* is governed by the demand and supply of the traded commodities (processing and communication capacities) and the competition created between the buyers (tasks) and the sellers (workstations), respectively. The work provides a basis for an elaborate study on realistic concurrent applications and workstation networks. It underlines the development of simple, faster, and flexible solutions to load distribution over clusters of workstations, inspired by the market rules.

1 Introduction

Advances in computer and networking technology have led abundance of computing power in the form of interconnected powerful workstations. These are not really intended for parallel or distributed computing. The computing systems that were intended for parallel computing are, comparatively, closely coupled, although the memory is usually distributed and the communication is message based. In this study the emphasis is put on utilisation of the first group of distributed computing environments, through balanced load distribution using market rules.

The workstations, although often on a network, are meant for the principle user whose desk or office it is located. However, it is evident that the principle user utilises only a fraction of own workstation capacity [7]. There is an obvious need for improving the utilisation of interconnected workstations. The problem, in its general form, is highly mathematical and its optimum solutions are computationally intractable, except for very restricted cases [2].

The objective of this work is to apply competition based market rules to general load balancing in distributed systems; in this presentation, the term load-balancing is used interchangeably with load-distribution. The distributed computing system (DCS) is a network of workstation

*Appears in the proc. of the 9th International Sym. on Computer and Information Sciences, Antalya, 1994.
†On sabbatical leave from King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia.

clusters (from now on referred to as ws-clusters). Each ws-cluster is a network of connected workstations. There is no special provision about the interconnection scheme of the workstations or inter-task communication requirements. In fact, irregular and random interconnections are more realistic as they include all other schemes.

The concurrent applications are represented by a connected graph of tasks. The tasks are related to each other by their communication requirements in terms of unit data (bytes), the workstations are related to each other by the price of transmission per unit data. In market terms, tasks are buyers of the commodities: processing and transmission power. The workstations are sellers of these commodities. The buyers should eventually bid to get the best possible deal in terms of their power of payment and urgency for completion in time. The sellers should eventually get the best possible price paid for the commodities that they auction. In doing so, both the tasks and the workstations may compete with other tasks and workstations, respectively. In principle, collaborations between friendly tasks and friendly workstations are possible. In that case, the competition exists between the ws-clusters, clusters of tasks (grains) in an application, or different applications.

2 Related work

Various studies have been conducted on the computational market and its application to load distribution. In [4], *microeconomic algorithms for load balancing in distributed computer systems* is studied. Here the tasks are assumed independent, i.e., with no communication between them. Each task receives a fund upon entry to the system. This fund (money) is used to bid for the resources needed, according to a preference rule. A preference of one processor to another can be based on service time only, price only, or the both. On the other hand, the processors auction their processing and communication resources, with an objective of maximising their profit. They have found that their microeconomic algorithms can achieve effective load distribution compared to no load distribution. Another study known as *Enterprise: a market-like task scheduler for distributed computing environments* was conducted in [6]. In this study, the tasks(clients) announce their requirements asking for bids, the idle processors (contractors) bid, in response. The tasks, in turn, choose the contractor with the best bid. Again the work involves independent tasks, and it is really not a market driven approach. There is no market like price control mechanism that considers supply-demand element in the bidding process.

In [8], a study on *distributed computational economy (Spawn)* is conducted. It is meant to support concurrent applications as well as independent tasks. The processors auction the next available slice of processing time. Depending on the strategy, an auction may consider giving discounts, or raise the prices. The concurrent application is represented by a tree of tasks. Each task is funded from the root, to bid in the auction. In their study, no provisions are given for the communication requirements between the tasks, the spawning considers tree-structured computations. The experiments that are conducted are based on Monte Carlo applications which are suitable for decomposition into a number of tasks, if so required.

3 Load distribution through competition

In this study, the concurrent applications are allowed to split or merge according to granularity requirement of the system. The cumulative processing and transmission capacity of the DCS is used to determine the split or merge option. To the best of our knowledge no study has considered this approach, although, in [1] and [3] the communication intensity between the tasks and

the system topologies have been used to partition the concurrent applications, by conventional methods. Also, here, no particular interconnection scheme is imposed on application and the DCS, in contrast to most of the earlier studies.

The applications are led through two phases: partitioning and mapping. The partitioning phase includes two stages: granulation and merge. The granulation stage involves computing the proportion of the application and the number of grains (task clusters) for each ws-cluster; the merge stage groups the tasks into grains. In doing so, the ws-cluster size; processing, interference, and transmission prices incurred by the individual workstations need to be considered. Following the partitioning phase, each selected ws-cluster will have a number of grains to be properly mapped to the constituting workstations. The mapping phase produces the best possible distribution based on the price determined by market rules. It, too, involves two stages: initial assignment and dynamic grain exchange between the workstations to improve the initial assignment. The components of the price involve processing, communication, and other factors that may represent market rules, which would normally show variations because of the supply and the demand for the resources concerned.

Let the application be represented by a triple $\Theta = (T, E, C)$ where T is a set of tasks, E is the set of task execution times, and C is the set of inter-task communication requirements. The DCS is a set of clusters, $\Gamma = \{\Gamma_i | i \in n\}$. A ws-cluster is a set of workstations, $\Gamma_i = \{W_{i,j}, | i \in n, j \in m_i\}$. The workstations are related to each other by the price of unit data transmission between them. Obviously, the price would be infinity if they were not related at all. The following list documents all the symbols used in the rest of the paper:

C_{ij}	Communication requirement (no. of bytes) between two tasks
C^T	Total communication requirement among tasks of an application
E_i	Task Execution requirement (no. of instructions)
E^T	Total task execution requirement of an application
N^T	Total number of tasks
N_s	Total number of clusters
M_i^a	No. of available workstations (ws) in a ws-cluster
P_i^p	unit processing price of a ws-cluster
P_i^t	unit transmission price of a ws-cluster
$P_{i,j}^e$	unit transmission price between two ws in different clusters
q_i	No. of grains in a ws-cluster
r_i	Total grain-size in a ws-cluster
t_p^u	Maximum expected processing time of an application
t_p^l	Minimum expected processing time of an application
t_t	Maximum expected transmission time of an application
t_e	Maximum expected inter-cluster transmission time
X_i	ws processing capacity (instr./ sec.) offered by a ws-cluster
Y_i	ws transmission capacity (bytes/ sec.) offered by a ws-cluster
$Z_{i,j}$	inter-cluster transmission capacity (bytes/ sec.) offered by two ws-clusters

3.1 Granulation of the Application

A grain consists of one or more tasks. Each grain is run on a single workstation. The grain-size is an indication of the capacity offered by the hosting workstation. The higher the capacity, the larger will be the grain-size. The algorithm shown in Figure 1 computes the total grain-size (r_b) and the number of grains (q_b) for a ws-cluster b. The prelude to the algorithm is to compute the normalised total price for each ws-cluster, which is a function of processing and communication capacities of the cluster. The ws-clusters (Γ^s) are sorted in ascending order of their normalised total prices as indicated at the bottom end of the algorithm.

First, for each ws-cluster b (initially, b=1), the algorithm chooses q_b, between the lower and upper bounds imposed by the application and workstation execution data. Second, it checks

Cluster_Grain (b, r_b) : initially the cluster number b=1, the grain size $r_1 = 1$

Begin

$M_b^l \longleftarrow \lceil \frac{E^T}{X_b \cdot t_p^u} \cdot r_b \rceil$

$M_b^u \longleftarrow \lceil \frac{E^T}{X_b \cdot t_p^l} \cdot r_b \rceil$

$q_b' \longleftarrow M_b^u \cdot f(P_b^p)$

While $[q_b' > (\frac{N^T \cdot Y_b \cdot t_t}{C^T} \cdot g(P_b^t))]$

If $(q_b' > M_b^l)$

$q_b' \longleftarrow q_b' - 1$

Else

Termination : fail to satisfy execution requirements

If $(q_b' > M_b^a)$

If $(b = N_s)$ or $[q_b' > (\frac{N^T \cdot Z_{b,b+1} \cdot t_e}{C^T} \cdot h(P_{b,b+1}^e))]$

Termination : fail to satisfy execution requirements

Else

$r_{b+1} \longleftarrow r_b \cdot (1 - \frac{M_b^a}{q_b'})$

$r_b \longleftarrow r_b \cdot \frac{M_b^a}{q_b'}$

$q_b \longleftarrow M_b^a$

Cluster_Grain $(b + 1, r_{b+1})$

Else

$q_b \longleftarrow q_b'$

End

Definition of Symbols :

$\Gamma^s = \{\Gamma_i^s | i \in n\}$, and Γ_i^s are arranged in sorted normalised total price order :

$$(\frac{E^T}{X_i} \cdot P_i^p + \frac{C^T}{Y_i} \cdot P_i^t) \leq (\frac{E^T}{X_{i+1}} \cdot P_{i+1}^p + \frac{C^T}{Y_{i+1}} \cdot P_{i+1}^t)$$

$f(P_b^p)$, $g(P_b^t)$, and $h(P_{b,b+1}^e)$ are the utilisation rate of average processing power, transmission capacity, and inter-cluster transmission capacity, and $P_{b,max}^p$, $P_{b,max}^t$, and $P_{b,max}^e$ are the maximum prices of the respective resources.

$$f(P_b^p) = \frac{P_b^p}{P_{b,max}^p}, \;\; g(P_b^t) = \frac{P_b^t}{P_{b,max}^t}, \;\; h(P_{b,b+1}^e) = \frac{P_{b,b+1}^e}{P_{b,max}^e}$$

Figure 1: **Granulation Algorithm.**

whether the inter-workstation transmission requirement is also satisfied. Since the average transmission cost is proportional to q_b, the transmission constraint can be represented as $Y_b \cdot g(P_b^t) \geq \frac{C^T \cdot q_b}{t_t \cdot N^T}$. If the transmission requirement is not satisfied, the value of q_b will be reduced, otherwise it will check whether q_b is greater than the number of available workstations in the ws-cluster. If the processing and transmission capacities are insufficient, the algorithm will split the application into grains of appropriate sizes to be allocated to this cluster. Grains can be assigned to other ws-clusters only if the inter-cluster transmission requirement is satisfied. The algorithm runs recursively until the entire application is split.

3.2 Merging Tasks into a Grain

The second stage is to merge tasks into q_b^T grains for each Γ_b^s according to the r_b and q_b. The merging algorithm, described in Figure 2, starts with selection of nuclei for each grain. The nuclei are the most demanding tasks in terms of processing and transmission costs (Φ). Each

nucleus may merge with a number of non-nuclei tasks. With each merge, the processing and transmission costs of the merging non-nuclei tasks are added to those of the nucleus.

The merging decision is determined by the value of, so called, decoupling force (df) between a nucleus and a non-nuclei task. The df between t_i and t_j, consists of two components as shown in Definition of Symbols section of the the merging algorithm in Figure 2. The first component (term) indicates the difference in processing cost. The second term is the difference in transmission cost. The interference costs account for the cost of incompatibility of task pairs [5]. I^p is called the *processor-based interference cost* incurred by process switching and synchronisation; I^{c1} is the interference cost incurred by interprocess communication. When two communicating tasks are executed on the same processor, the overhead in multiplexing/demultiplexing the aggregate communication data should also be considered as illustrated in (Figure 3). I^{c2} represents this overhead which possesses the following property :

$$\sum_{t}^{n_i+n_j-2} I^{c2}(i,t) \geq \sum_{t}^{ni} I^{c2}(i,t) + \sum_{t}^{nj} I^{c2}(j,t)$$

The tasks assigned to different processors are assumed not to compete for the same communication facility. Conversely, if tasks are assigned to the same workstation, they also share the communication facilities, such as the communication processor. Higher values of df represent higher tendency for separating the tasks, thus, favouring the parallel execution. Therefore, merging two tasks with the minimum values of df is to maximise the benefit of concurrency, under the processing and transmission constraints of the ws-clusters. If the value of df is negative, then it is preferable to execute the tasks in the same workstation.

Out of the q_b^T grains remained to be considered at this point, only q_b of them, with the highest values of Φ, are selected as the grains that can be executed on Γ_b^s. If the value of df between any pair of the q_b^T grains is negative, the value of q_b^T will be reduced and the merging algorithm will be restarted on Γ_b^s. The merging algorithm will be repeated until all the grains of an application have been allocated to the ws-clusters, Γ^s.

4 Mapping

After the merging process, the mapping algorithm that maps the allocated grains into workstations will be executed on one of the workstations in each ws-cluster, called *cluster controller*. The algorithm (Figure 4) has two parts: initial mapping and dynamic exchange. In the first part, the grain with the maximum total cost of processing and transmission is assigned to the workstation with the minimum total price. The processing price of the workstation $(P_{b,x}^p)$ is then updated according to the proportion of consumption as in Eq.1.

$$P_{b,x}^p = P_{b,x}^p \cdot (1 + \frac{X_b}{X_b^T}) \qquad \text{(Eq.1)}$$

Where X_b^T is the total processing capacity of a workstation in Γ_b^s.

After the first assignment, the next adjacent grain with the maximum total cost is assigned to the workstation with the minimum total price. Besides the processing price, the link transmission prices $(P_{b,\alpha\beta}^t)$ along the path between the workstations to which the two adjacent grains are assigned are updated as in Eq.2. This mapping is repeated until all the grains have been mapped.

$$P_{b,\alpha\beta}^t = P_{b,\alpha\beta}^t \cdot (1 + \frac{Y_b}{Y_b^T}) \qquad \text{(Eq.2)}$$

Begin
For every $\Gamma_b^s \in \Gamma^s$ and $r_b > 0$,
Select q_b^T tasks from T to be nuclei with highest values of Φ
For every $t_i \in T$ except the nuclei
For every nucleus t_j
Calculate decoupling force $df(i,j)$
Merge t_i into t_j with the minimum value of $df(i,j)$,
(break ties by smaller value of E_j)
Select q_b task clusters with highest values of Φ allocated in Γ_b^s
End

Definition of Symbols :

$$
\begin{aligned}
q_b^T &= \frac{q_b}{r_b} - \sum_{i=1}^{b-1} q_i \\
\Phi &= E_i \cdot P_b^p + (\sum_j C_{ij}) \cdot P_b^t \\
df(i,j) &= \{[(E_i + E_j) + I^p(i,j)] - \text{Max}(E_i, Ej)\} \cdot P_b^p \\
&\quad + \{[\sum_l^{n_i+n_j-2} I^{c2}(i,l) + I^{c1}(i,j)] \\
&\quad - [\sum_l^{ni} I^{c1}(i,l) + \sum_l^{nj} I^{c2}(j,l) + C_{ij}]\} \cdot P_b^t
\end{aligned}
$$

where I^p is processor-based interference cost,
I^{c1} is interference cost due to intertask communication,
I^{c2} is interference cost due to multiplexing and demultiplexing,
n_i is the number of communication links of task t_i

Figure 2: **Merging Algorithm**

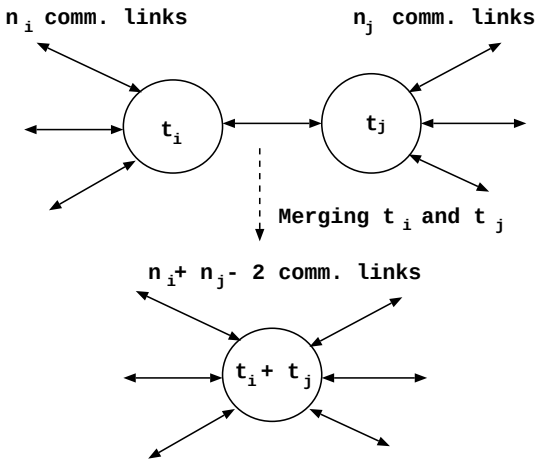

Figure 3: **Merging of two tasks**

Initial assignment

Begin
Select a grain t_i with Max$(P_b^p \cdot E_i + (\sum_j C_{ij}) \cdot P_b^t)$
Assign t_i to $W_{b,x} \in \Gamma_b^s$ with Min$(P_{b,x}^p \cdot E_i + (\sum_j C_{ij}) \cdot$ Max$(P_{b,xz}^t))$,
where $W_{b,z} \in \Gamma_b^s$ is an adjacent nodes of $W_{b,x}$
Update processing price of $W_{b,x}$ by applying Eq.1
While there are any grain t_j which have not been assigned
Select t_j adjacent to t_i with Max$(P_b^p \cdot E_i + C_{ij} \cdot P_b^t)$
Assign t_j to $W_{b,y} \in \Gamma_b^s$ with Min$(P_{b,y}^p \cdot E_j + C_{ij} \cdot P_{b,xy}^t)$
Update processing prices of $W_{b,y}$ by applying Eq.1
Update link transmission prices along $\langle W_{b,x}, W_{b,y} \rangle$
to which t_i and t_j are assigned by applying Eq.2
If all the adjacent grains of t_i have been assigned,
Select the ws to which t_i is assigned as $W_{b,x}$,
and the assigned grain with Max$(P_b^p \cdot E_i + (\sum_j C_{ij}) \cdot P_b^t)$ as t_i
End

Dynamic Exchange

Begin
For every $W_{b,x} \in \Gamma_b^s$,
While there is any $W_{b,y} \in \Gamma_b^s$ s.t. $\Phi_{i,x} > (\Phi_{i,y} + \epsilon)$,
If Max$(\Psi_{x,y}) > \epsilon$
Perform task exchange between $W_{b,x}$ and $W_{b,y}$
Else
Termination
Update processing prices of the exchanged tasks by applying Eq.1
Update link transmission prices along $\langle W_{b,x}, W_{b,y} \rangle$ by applying Eq.2.
End

Definition of Symbols :

$$\Phi_{i,x} = E_i \cdot P_{b,x}^p + \textstyle\sum_k (C_{ik} \cdot P_{ik,xz}^v)$$

$$\Psi_{x,y} = (\Phi_{i,x} - \Phi_{j,x}) + (\Phi_{j,y} - \Phi_{i,y})$$

where $\Phi_{i,x}$ is the total price of executing t_i in $W_{b,x}$,
$P_{ik,xz}^v$ is the price on the path $\langle W_{b,x}, W_{b,z} \rangle$, if t_i was assigned to $W_{b,x}$,
t_k is the adjacent grain to t_i,
$\Psi_{x,y}$ is the gain incurred, if the tasks t_i on $W_{b,x}$ and t_j on $W_{b,y}$ are switched.

Figure 4: **Mapping Algorithm**

Where $W_{b,\alpha}$ and $W_{b,\beta}$ are the adjacent workstations along the path between workstations $W_{b,x}$ and $W_{b,y}$ to which t_i and t_j are assigned respectively, and Y_b^T is the total transmission capacity of a workstation in Γ_b^s.

The second part of the algorithm involves a dynamic exchange to improve the initial mapping. First, the cluster controller searches for any differences of total prices for grain execution that is greater than the amount of the grain exchange overhead (ϵ). Second, if the maximum gain in a grain exchange ($\Psi_{x,y}$) is greater than ϵ, then the grain exchange will be performed. After each exchange, the processing and transmission prices related to the exchange are updated accordingly. The algorithm repeats until all the differences of the total prices are not greater than ϵ, or any maximum gain is not greater than ϵ.

5 Conclusion

In this work, an integrated solution approach is developed for load balancing in a DCS environment that involves concurrent applications and clusters of workstations, inspired by the market rules. The main aim is to handle the problem in its comprehensive framework, but with simplification and flexibility provided by the market rules which are often translated into price. The algorithm could allow dynamic adjustments, effected by the changes in the ws-clusters and/or the applications. All required is to incorporate the changes in the new auction and bidding prices of the commodities (processing and communication capacities). The formulation developed, which takes the price as its heart, acknowledges the feasibility of the approach.

The presentation has provided a framework for future work which is planned to involve implementation and analysis, regarding various concurrent applications and ws-clusters.

References

[1] R. Agrawal, and H. V. Jagadish, 'Partitioning techniques for large-grained parallelism,' IEEE Trans. on Computers, Vol.37, No.12, 1988, pp.1627-1634.

[2] S.H. Bokhari, 'A Shortest Tree Algorithm for Optimal Assignments across Space and Time in a Distributed Processor Systems,' IEEE Trans. on Software Engineering, Vol.SE-7, No.6, 1981, pp.583-589.

[3] M. Bozyigit, U. Kalaycioglu, and M. Melhi,'Load balancing in dense distributed systems,' Proceedings of the 4th ISCIS, Cesme, Vol. 1, pp.345-361, October 1989.

[4] D. Ferguson, Y. Yemini, C. Nikolaou, 'Microeconomic algorithms for load balancing in distributed computer systems,' Proc. 8th International Conf. on Distributed Computing Systems, 1988, pp.491-499.

[5] V.M. Lo, 'Heuristic algorithms for task assignment in distributed systems,' IEEE Trans. on Computers, Vol.37, No.11, 1988, pp.1384-1397.

[6] T.W. Malone, R.E. Fikes, K.R. Grant, and M.T. Howard, 'Enterprise: a market-like task scheduler for distributed computing environments,' in The Ecology of Computation, Huberman B.A., Ed. Amsterdam:North-Holland, 1988, pp.177-205.

[7] M.W. Mutka, and M. Livny, 'Profiling workstations' available capacity for remote execution,' Performance'87, Proceedings of the 12th IFIP WG &.3 Symposium on Computer Performance, Brussels, Belgium, December, 1987.

[8] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart, and W.S. Stornetta, 'Spawn: a distributed computational economy,' IEEE Trans. on Software Engineering, Vol.SE-18, No.2, 1992, pp.103-117.